High Performance Multivariate Visual Data Exploration for Extremely Large Data

Oliver Rübel[1,5,6], Prabhat[1], Kesheng Wu[1], Hank Childs[2], Jeremy Meredith[3], Cameron G.R. Geddes[4], Estelle Cormier-Michel[4], Sean Ahern[3], Gunther H. Weber[1], Peter Messmer[7], Hans Hagen[6], Bernd Hamann[1,5,6] and E. Wes Bethel[1,5]

Abstract

One of the central challenges in modern science is the need to quickly derive knowledge and understanding from large, complex collections of data. We present a new approach that deals with this challenge by combining and extending techniques from high performance visual data analysis and scientific data management. This approach is demonstrated within the context of gaining insight from complex, time-varying datasets produced by a laser wakefield accelerator simulation. Our approach leverages histogram-based parallel coordinates for both visual information display as well as a vehicle for guiding a data mining operation. Data extraction and subsetting are implemented with state-of-the-art index/query technology. This approach, while applied here to accelerator science, is generally applicable to a broad set of science applications, and is implemented in a production-quality visual data analysis infrastructure. We conduct a detailed performance analysis and demonstrate good scalability on a distributed memory Cray XT4 system.

Keywords: data mining, visual data analysis, accelerator modeling, parallel visualization, large data visualization, temporal visualization, temporal data analysis

1 INTRODUCTION

To support scientific knowledge discovery and hypothesis testing on datasets of ever increasing size and complexity, our work here focuses on combining and extending two different but complementary technologies aimed at enabling rapid, interactive visual data exploration and analysis of contemporary scientific data. To support highly effective visual data exploration, we have adapted and extended the concept of parallel coordinates, in particular binned or histogram-based parallel coordinates, for use on visual exploration of very large data, multivariate datasets. In the context of visual data exploration and hypothesis testing, the parallel

1. Computational Research Divisition, Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720, USA.
2. Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550.
3. Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, TN 37831, USA.
4. LOASIS program of Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720, USA.
5. Institute for Data Analysis and Visualization (IDAV) and Department of Computer Science, University of California, Davis, One Shields Avenue, Davis. CA 95616, USA.
6. International Research Training Group "Visualization of Large and Unstructured Data Sets Applications in Geospatial Planning, Modeling, and Engineering," Technische Universität Kaiserslautern, Erwin-Schrödinger-Straße, D-67653 Kaiserslautern, Germany.
7. Tech-X Corporation, 5621 Arapahoe Ave. Suite A, Boulder, CO 80303.

Manuscript received 14 April 2008.

coordinates display and interaction mechanism serves multiple purposes. First, it acts as a vehicle for visual information display. Second, it serves as the basis for the interactive construction of compound Boolean data range queries. These queries form the basis for subsequent "drill down" or data mining actions. To accelerate data mining, we leverage state-of-the-art index/query technology to quickly mine for data of interest as well as to quickly generate multiresolution histograms used as the basis for the visual display of information. This combination provides the ability for rapid, multiresolution visual data exploration.

We apply this new technique to large, complex scientific data created by a numerical simulation of a laser wakefield particle accelerator. In laser wakefield accelerators, particles are accelerated to relativistic speeds upon being "trapped" by the electric fields of plasma density waves generated by the radiation pressure of an intense laser pulse fired into the plasma. These devices are of interest because they are able to achieve very high particle energies within a relatively short amount of distance when compared to traditional electromagnetic accelerators. The VORPAL [15] simulation code is used to model experiments, such as those performed at the LOASIS facility at LBNL [7], and is useful in helping to gain deeper understanding of phenomena observed in experiments, as well as to help formulate and optimize the methodology for future experiments.

Laser wakefield simulations model the behavior of individual particles as well as the behavior of the plasma electric and magnetic fields. Output from these simulations can become quite large: today's datasets, such as the ones we study here, can grow to be on the order of 200GB per timestep, with the simulation producing hundreds of timesteps. The scientific challenge we help address in this study is first quickly identifying particles in simulation data that have undergone wakefield acceleration, trace them backwards in time to understand acceleration dynamics, and perform both visual and quantitative analysis on the set of accelerated particles.

One scientific impact of our work is that we have vastly reduced the duty cycle in visual data exploration and mining. In the past, accelerator scientists would perform the "trace backwards" step using scripts that performed a search at each timestep for a set of particles. Runtimes for this operation were on the order of hours. Using our implementation, those runtimes are reduced from hours to seconds.

The specific new contributions of this work are as follows:

- We present a novel approach for quickly creating histogram-based parallel coordinates displays. These displays serve to convey information as well as the interface for the interactive construction of compound, multivariate Boolean range queries. The new approach leverages state-of-the-art index/query technology to achieve very favorable performance rates.
- We apply this new approach to solve a challenging scientific data understanding problem in accelerator modeling. This new approach performs particle tracking in a few seconds as compared to hours when using a naive script.
- We examine and report the performance of our approach using a modern HPC platform on a very large ($\approx$1TB) and complex scientific dataset. We demonstrate that our approach has excellent scalability characteristics on a distributed memory Cray XT4 system.

The rest of this paper is organized as follows. First, we review relevant background work in Section 2, which covers a diverse set of topics in visual information display, index/query technology, and high performance visual data analysis software architectures. Next in Section 3, we present the architecture and implementation of our approach. This approach is then applied

to solve a challenging scientific data understanding problem in the field of laser wakefield accelerator modeling in Section 4. We evaluate the performance of our system and present performance results in Section 5. Finally, we conclude in Section 6 with a summary and suggestions for future work.

2 Related Work

2.1 Information Display – Parallel Coordinates

Parallel coordinates, proposed by Inselberg [11] and Wegman [23], are a common information visualization technique for high-dimensional data sets. In parallel coordinates, each data variable is represented by one axis. The parallel coordinates plot is constructed by drawing a polyline connecting the point where a data record's variable values intersect each axis. This type of plot is expensive in the sense that many pixels are required to represent a single data record. In this form of display, there is substantial data occlusion by the many polylines required to display all records from a large dataset. Besides visualization of data subsets via selection and data segmentation, visualizations based on aggregation of data subsets have been proposed for visualization of very large data. An overview of modern parallel coordinates is provided in [13, 12].

Fua et al. [6] proposed the use of hierarchical parallel coordinates. They use hierarchical clustering to create a multiresolution view of the data, thereby allowing data exploration at varying levels of detail. Johannson et al. [14] used clustering to determine the inherent structure of the data, and displayed that structure with high-precision textures using different texture transfer functions. Novotný then used a binning based on a *k-means* clustering approach for creating an aggregate parallel coordinates visualization [16]. All these approaches are well-suited for presenting static data but are not very well suited for temporal data since defining temporally consistent clusters is non-trivial and computationally expensive.

Our histogram-based parallel coordinates approach extends the work of Novotný and Hauser [17], who proposed the use of binned parallel coordinates as an output oriented approach towards parallel coordinates. The main limitation of this approach is that in order to achieve interactive rendering speed, their method precomputes all possible 2D histograms with a fixed resolution of 256×256 and regular, equal-sized histogram bins. To implement different level of detail views, bins in the pre-computed histograms were merged, reducing the number of bins by half in each drill down step. While this strategy is efficient, it has several limitations. First, it does not allow for smoothly drilling into finer-resolution views of the data. Second, it supports presentation of binned views of the entire dataset, but does not support user-defined data subsetting. Finally, the fixed 256×256 histogram resolution exhibits significant aliasing when zooming in on narrow variable ranges in the parallel coordinates plot. Their approach uses binned parallel coordinates for "context views" and traditional, polyline-based parallel coordinates for "focus views." These focus views may still contain a substantial number of data records and will suffer from extensive occlusion as a result.

2.2 High Performance Index/Query for Data Mining

The data access patterns for data mining and analysis applications tend to be markedly different than those for transaction-based applications. Transaction-based applications tend to read and then update records in a database. In contrast, data mining and analysis applications tend to be read-only in their access pattern. The database indexing technology that is best suited for this type of data access is known as the bitmap index [4, 18]. The core idea of a bitmap index is to use a sequence of bits to mark the

positions of records satisfying certain conditions. Searching bitmap indices for data records that match a set of query conditions is performed efficiently using Boolean operations on bit vectors.

In uncompressed form, these bitmap indices may require too much space for a variable that has many distinct values, such as position or momentum of a particle. Several techniques have been proposed to improve the efficiency of the bitmap indexes on such variables [3, 26]. We make use of a bitmap index software called FastBit[1]. It implements the fastest known bitmap compression technique [25, 27], and has been demonstrated to be effective in a number of data analysis applications [2, 24]. In particular, it has a number of efficient functions for computing conditional histograms [20], which are crucial for this work.

When a variable has high cardinality, such as a floating-point value, each column in a bitmap index contains on/off bits for a range of values rather than a single data value. Conceptually, this approach is similar to a histogram; each column of the bitmap is a bin containing a range of data values. FastBit offers a number of different options for creating bitmap bins. We observe that users typically specify conditions with relatively low precision values, such as pressure less than 1×10^{-5} or momentum greater than 2.5×10^{8}. In the first case, the constant 1×10^{-5} is said to have 1-digit precision and the constant 2.5×10^{8} have 2-digit precision. We instruct FastBit to round data values to a user specified precision when deciding to which bins they belong. Such binning ensures that the above range conditions are resolved efficiently.

2.3 High Performance Query-Driven Visualization

The combination of high performance index/query with visual data exploration tools was described by Stockinger et al. [22] using the term "Query-driven visualization." That work focuses on comparing the performance of such a combination with state-of-the-art, tree-based searching structures that form the basis for a widely-used isocontouring implementation. Their work shows that this approach outperforms tree-based search structures for scalar variables, and also points out that all tree-based index/search structures are not practical for large, multivariate datasets since tree-based structures suffer from the "Curse of Dimensionality" [1]. The basic idea there is that storage complexity grows exponentially as one adds more and more search dimensions (e.g., more variables to be indexed/searched).

These concepts were later extended to the analysis of massive collections of network traffic data in two related works. First, the notion of performing network traffic analysis using statistics (e.g., histograms) rather than raw data led to a methodology that enabled exploration and data mining at unprecedented speed [2]. That study showed using these concepts to rapidly detect a distributed scan attack on a dataset of unprecedented size – 2.5 billion records. There, users are presented with an interface consisting of histograms of individual variables, and then they formulate a complex query via a process that is essentially a histogram "cross product." The process of data mining was subsequently accelerated through a family of algorithms for computing conditional histograms on SMP parallel machines [21].

2.4 High Performance Visual Data Analysis

Childs et al. [5] demonstrated a data processing and visualization architecture that is capable of scaling to extreme dataset sizes. This software system, VisIt[2], has been shown to scale to 10s of billions of data points per timestep and runs in parallel on nearly

[1] The FastBit software is available from `https://codeforge.lbl.gov/projects/fastbit/`.

[2] VisIt is available from `https://wci.llnl.gov/codes/visit/`

all modern HPC platforms. In reference to our work, VisIt employs a contract-based communications system that allows for pipelining and I/O optimization, reducing unnecessary processing and disk access. It is this extensible contract system that we utilize here to generate histograms for visual data exploration. A specific optimization that we employ is VisIt's concept of a set of Boolean range queries. This set may be communicated between data processing modules in an out-of-band fashion, allowing downstream filters to limit the scope of work of upstream filters.

3 SYSTEM DESIGN

We now describe the the general work flow for analyzing very large data and explain the design of our system. Subsequent sections provide more detailed explanations on using histograms to generate parallel coordinate views effectively (Section 3.1) and on how our system performs data selection (Section 3.2).

Before analysis, we need to load the data of interest. Due to the sheer size of the data, it is often prohibitively expensive, if not impossible to load the entire dataset into main memory. To address this problem, we initially read only meta-data, which includes information such as number of data dimensions, number and name of data variables, number of data records, and the number of time steps.

We have adapted histogram-based parallel coordinates as the principal interface for data selection and exploration. Data selection is a well-established concept in visual data exploration as evidenced by the visual information seeking mantra: "overview first, zoom and filter, then details-on-demand." It is particularly applicable for exploring extremely large data since it limits the amount of information that needs to be considered at a given time. In our implementation, the entire data set (or a data subset) is rendered in the context of the parallel coordinates view that provides the user with information about trends in the data. After defining data subsets of interest, we then conduct a much more detailed and focused analysis on the smaller subset. Using sliders attached to each parallel axis, a user can quickly define ranges of interest in any data dimension or variable. We subsequently render these significantly smaller data set portions in high resolution (the focus view), which is displayed concurrently with the low resolution view (the context view). As we will describe later in Section 3.1, we can use the same efficient histogram-based rendering for the focus of a parallel coordinates plot.

When dealing with extremely large data we need to be able to perform data selection efficiently. Within our system we make use of two different types of data selection: (i) selection based on multivariate thresholding, and (ii) selection-based identifiers. Once a user defines "interesting" values of data with multivariate thresholding, we use FastBit to locate the identifiers of the "interesting" data records. In our example, the identifiers represent particle identification. Next, we take the list of particle identifiers and again use FastBit to trace those particles over time. We describe accelerated data selection using FastBit in more detail in Section 3.2.

To perform a detailed data analysis, it is often useful to correlate different views with one other. In addition to the parallel coordinates view, which serves both as a data visualization and multivariate range selection mechanism, we allow users to display interesting data in a variety of plot types, such as pseudocolor or scatter plots. We demonstrate the usefulness of our system in Section 4, which presents an analysis of laser wakefield acceleration simulation data.

3.1 Histogram based Parallel Coordinates

Parallel coordinates provide a very effective interface for defining multi-dimensional queries based on thresholding. Using sliders attached to each axis of the parallel coordinates plot, a user defines range thresholds in each displayed dimension. By rendering the user selected data subset (the focus) in front of a plot of the entire dataset or a larger data subset(the context), the user receives immediate feedback about general properties of the selection. Data outliers visually stand out as single or small groups of lines diverging from the main data trends. Data trends appear as dense groups of lines (bright colored bins in our case). A quick visual comparison of the selected subset with the complete data helps to convey understanding about similarities and differences between the two. In practice, parallel coordinates have disadvantages when applied to very large datasets. First, each data record is represented with a single polyline that connects each of the parallel coordinates axes. As data size increases, the plot becomes more cluttered and difficult to interpret. Also, data records drawn later will occlude information provided by data records drawn earlier. Worst of all is that this approach has computational and rendering complexity that is proportional to the size of the dataset. As data sizes grow ever larger, these problems become intractable.

To address these problems, we employ an efficient rendering technique based on two-dimensional (2D) histograms. Rather than viewing the parallel coordinates plot as a collection of polylines, one per data record, we approach rendering by considering instead the relationships of all data records between pairs of parallel coordinate axes. That relationship can be discretized, and rendered, as a 2D histogram. This idea was introduced in earlier work [19, 17].

As illustrated in Figure 2, we create a parallel coordinates representation based on 2D histograms by drawing one quadrilateral per non-empty bin, where each quadrilateral connects two data ranges between neighboring axes. As illustrated in Figure 1(a,b), histogram-based rendering overcomes the limitations of polyline-based rendering and reveals much more data detail when dealing with a large number of data records. In the following, we first describe how to quickly compute 2D histograms, and then explain in detail how to use these 2D histograms to efficiently render parallel coordinates. Having introduced the principle of histogram-based parallel coordinates, we then compare use of regular (equal width) and adaptive (equal weight) binned 2D histograms in the context of rendering parallel coordinate views.

Within VisIt, 2D histograms are computed directly in the file reader, which leverages FastBit for index/query operations as well as histogram computation. This approach has several major benefits within the context of very large, high performance visual data analysis. First, instead of having to read the entire data set and transfer it to the plot to create a rendered image, we limit internal data transfer and processing to a set of 2D histograms, which are very small when compared to the size of the source data. Second, data I/O is limited to those portions needed for computing the current 2D histogram, i.e., information on the relevant particles in two data dimensions. After computing a 2D histogram, we can then discard the data, thus decreasing the memory footprint. An added advantage is the fact that all the computationally heavy work is done at the beginning of the execution pipeline rather then at the end in the plot. Third, having the histogram computation be part of the file reader allows us to perform such computation at the same stage of processing as parallel I/O, which is one of the most expensive operations in visual data analysis. This approach provides the ability to achieve excellent parallel performance, as described later in Section 5.

When using 2D histograms for rendering parallel coordinates plots, we have access to additional information, which is not present when using traditional polyline-based methods. This extra information allows us to optimize various aspects of data

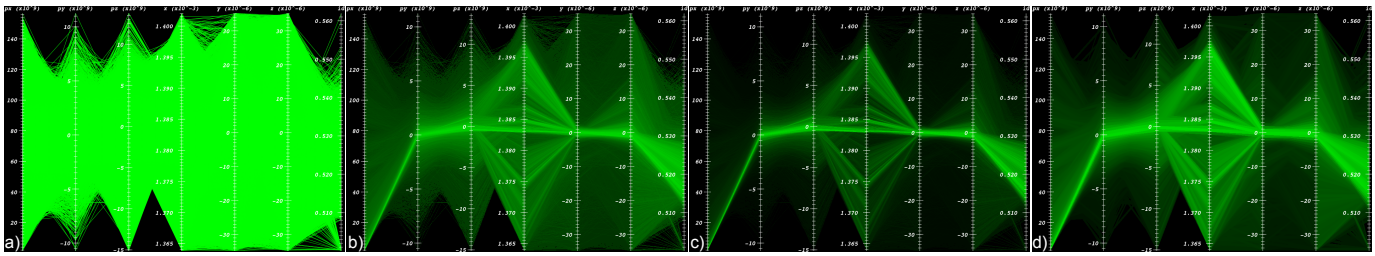

Fig. 1: Comparison of different parallel coordinate renderings of a subset of a 3D laser wakefield particle acceleration dataset consisting of 256,463 data records and 7 data dimensions. a) Traditional line based parallel coordinates. b) High-resolution, histogram-based parallel coordinates with 700 bins per data dimension. c) Same as in *b*, but using a lower gamma value *g* defining the basic brightness of bins. d) Same rendering as in *b* but using only 80 bins per data dimension. When comparing *a* and *b*, we see the histogram-based rendering reveals many more details when dealing with a large number of data records. As illustrated in *c*, by lowering gamma we can then reduce the brightness of the plot and even remove sparse bins, thereby producing a plot that focuses on the main, dense features of the data. By varying the number of bins we can then create renderings at different levels of detail.

visualization to convey more information to the user. We know, for example, the number of records contributing to specific variable ranges between parallel axes. We use brightness, for example, to reflect the number of records per bin, which leads to improved visual presentation. Assuming that denser regions are more important than sparse regions, we render bins in back-to-front order with respect to the number of records per bin $h(i, j)$. Figure 1(a,b) shows a direct comparison of the same data once rendered using traditional line based parallel coordinates and once using our histogram based rendering approach.

In order to further improve the rendering, we allow the user to define a gamma value g defining the basic brightness of the plot. As illustrated in Figure 1c, by lowering g the user can reduce the brightness of the plot or even remove sparse bins from the rendering thereby producing a much less cluttered visualization that focuses attention on the main, dense data features. Since our method is not constrained by a fixed histogram bin resolution, we can easily recompute histograms at higher resolution, or, using adaptive binning, produce visualizations at varying levels of detail. This feature is important for providing smooth drill-down into finer levels of detail in very large datasets and represents one of the major improvements of our approach over previous work. Figure 1d, shows as an example the same rendering as in Figure 1b, but using just 80 bins per data dimension. Another example is also provided later in Figure 3.

Previous histogram-based parallel coordinates work used histogram-based rendering for the context view and traditional line-based rendering for the focus view. One limitation of this approach is that the focus view may consist of a very large number of data records. We overcome this limitation by using a histogram-based approach for both, the context and focus views. This approach is feasible given the rapid rate at which we can recompute new conditional 2D histograms as explained in Section 5. The focus view is rendered on top of the context view using a different color to make the focus more easily distinguihable. Using histogram-based rendering for the focus view further has the advantage that we can render it at different levels of detail simply by specifying the number of bins per variable. Here, we can use a lower-resolution histogram for the context view and a very high-resolution histogram for the focus view, thereby supporting a high quality and smooth visual drill-down into the focus view.

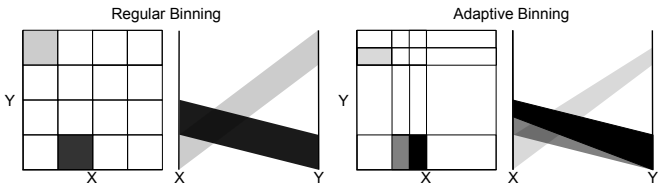

Fig. 2: Illustration showing rendering of parallel coordinates based on regular (left) and adaptive binned histograms (right). By using higher resolution in areas of extremely high density, an adaptive binning is able to represent the general data trends much more accurately.

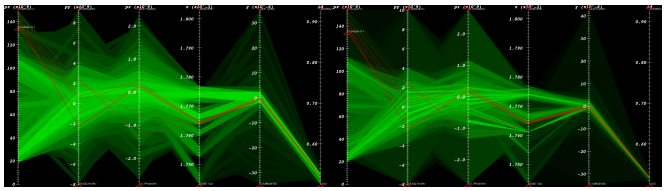

Fig. 3: Histogram-based parallel coordinates using $32x32$ regular binned histograms (left) and adaptive histograms (right). Compared to regular binning, the adaptive binning preserves more details in dense areas while discarding some details in sparse areas of the data. An adaptive binned plot may ease comparison of selections (red) with structures present in the data in low level of detail views.

Similar to rendering the focus on top of the context, we can also create a rendering of multiple timesteps in one parallel coordinates plot. To do so, we assign a unique color to each timestep and render the individual plots, each representing one timestep, on top of one other. As we will show later in Section 4, such a rendering can be helpful to identify the general temporal changes in the data. In practice, temporal parallel coordinates are most useful when analyzing some characteristic subset of the data.

Our approach offers another improvement over previous work, namely we can compute and render with adaptive, rather than uniform histogram bins. With adaptive binning, each bin of the histogram contains approximately the same number of data records, which may offer advantages in certain situations [10]. As an example, Figure 3 shows a comparison of data visualized using $64x64$ uniform versus adaptive histogram bins. In comparison to a uniform binning, adaptive binning discard some features in sparse areas of the data to preserve more information in dense areas. Adaptively binned histograms may ease comparison of selections with general data trends. As illustrated in Figure 2, when using adaptive binned histograms, a more generalized rendering is required to allow rectangles to connect different sized ranges on neighboring axes. Also, since the area $a(i,j)$ covered by each bin is no longer constant, we need to compute the brightness of each bin, and then assign rendering order based on the actual data density per bin $p(i,j) = \frac{h(i,j)}{a(i,j)}$, rather than directly on $h(i,j)$. Uniform binned histograms are in general very well suited for high resolution renderings of the data while adaptive binned histograms may be advantageous for low resolution renderings.

3.2 Data Selection

In our implementation, there is synergy between presentation of information to the user and specification of queries. The parallel coordinates plot serves to present context and focus data views to the user, and also serves as the mechanism for a user to specify a multivariate Boolean range query. In our example case study in Section 4, a multivariate range query might take the form of $px > 10^9$ && $py < 10^8$ && $y > 0$, which selects high momentum particles in the upper half of the beam. Similar conditions can be formulated to define arbitrary subsets of particles with interesting momentum and spatial characteristics.

Once the user specifies such a multivariate range condition, those conditions are passed back upstream in the system to the FastBit-enhanced HDF5 reader for processing, either to compute new histograms or to extract data subsets that match the query for downstream processing. In the case of data subsetting, FastBit will locate those data records that satisfy the query and then pass them along for downstream processing. For conditional histograms, FastBit will compute new histograms using the query conditions as well as a histogram specification: the number of bins and the bin boundaries.

Once an interesting subset has been identified, yet another form of query can be issued in order to identify the same data ranges at different points in time. This type of query is of the form $ID\ IN\ (id1, id2, ..idn)$, where there are n particles in the subset. Again, such queries can be processed efficiently by FastBit and only the relevant set of particles is extracted and then passed along to visual data analysis machinery. This processing step offers a huge performance advantage: a technique without access to the index information must search the entire dataset for particle identifier matches. By issuing an identifier query across the entire time sequence, we construct particle tracks, which reveal valuable information about how particles are accelerated over time.

4 Use Case

We now present a specific example where we apply our system to perform visual data analysis of 2D and 3D data produced by a laser wakefield particle accelerator simulation. These simulations model the effects of a laser pulse being applied to a hydrogen plasma. Similar to the wake of a boat, the radiation pressure of the laser pulse displaces the electrons in the plasma, and with the space-charge restoring force of the ions, this displacement drives a wave (wake) in the plasma. Electrons can be trapped and accelerated by the longitudinal field of the wake, forming electron bunches of high energy. Due to the large amount of particles required in order achieve accurate simulation results, it is not possible to simulate the entire plasma at once. As a result, the simulation is restricted to a window that covers only a subset of the plasma in x direction in the vicinity of the beam. The simulation code moves the window along the local x axis over the course of the run.

The simulation data in these examples contains information about the position of particles in physical space –x, y, z –, particle momentum – px, py, pz – and particle id at multiple timesteps. As a derived quantity, we also include the relative particle position in x direction within the simulation window $x_{rel}(t) = x(t) - max(x(t))$. There are more than $400,000$ and 90 million particles in the 2D and 3D datasets, respectively. The 2D data consists of 38 timesteps and has an overall size of about $1.3GB$, including the index structures. The 3D dataset consists of 30 timesteps and has an overall size of about $210GB$, including the index. Each 3D timestep has a size of about $7GB$, including the index, and about $5GB$ without the index. We perform a detailed analysis of the 2D dataset and then extend our analysis to the larger 3D dataset.

In order to gain a deeper understanding of the acceleration process, we need to address complex questions such as: i) which particles become accelerated; ii) how are particles accelerated, and iii) how was the beam of highly accelerated particles formed and how did it evolve [8]. To identify those particles that were accelerated, we are going to first perform selection of particles at a late time point of the simulation by using a threshold for the value for x-momentum, px. By tracing the selected particles over time we will then analyze the behavior of the beam during late timesteps (Section 4.1). Having defined the beam, we can analyze formation and evolution of the beam by tracing particles further back to the time where they entered the simulation and became

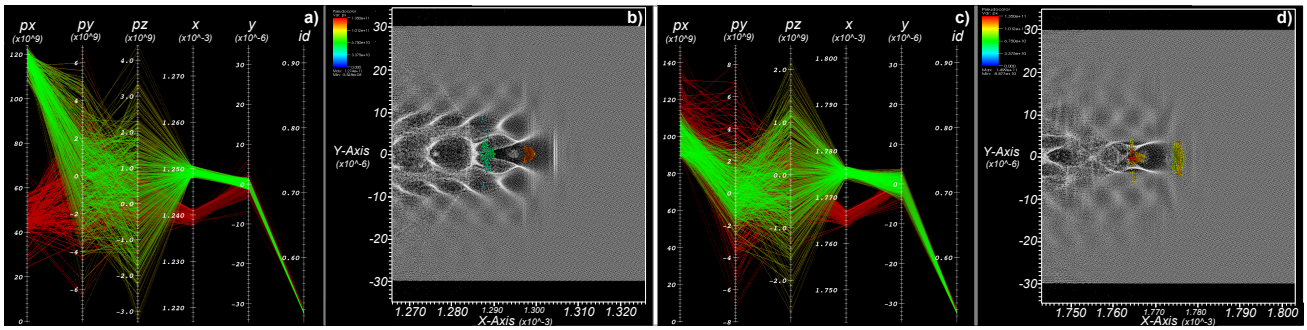

Fig. 4: a) Parallel coordinates and b) pseudocolor plot of the beam at $t = 27$. Corresponding plots c,d) at $t = 37$. The context plot, shown in red, shows both beams selected by the user after applying a threshold of $px > 8.872 * 10^{10}$ at $t = 37$. The focus plot, shown in green, indicates the first beam that is following the laser pulse. In the pseudocolor plots b) and d), we show all particles in gray and the selected beams using spheres colored according to the particle's x-momentum, px. The focus beam is the rightmost bunch in these images. At timestep $t = 27$, the particles of the first beam (green in figure a) show much higher acceleration and a much lower energy spread (indicated via px) than the particles of the second beam. At later times, the lower momentum of the first beam indicates it has outrun the wave and moved into decelerating phase, e.g at timestep $t = 37$.

injected into the beam (Section 4.2). In this context, we will use selection of particles, tracing of particles over time based on their ID's, as well as refinement of particle selections based on information from different time steps as the main analysis techniques.

As we will illustrate in this use case, the ability to perform selection interactively and immediately validate selections directly in parallel coordinates as well as other types of plots, such as pseudocolor or scatter-plots, enables much more accurate selection than previously possible. Tracing of particles over time then enables researchers to better understand evolution of the beam. With our visualization system the user can rapidly identify subsets of particles of interest and analyze their temporal behavior. For "small" one will usually perform all analysis on a regular workstation while for larger datasets VisIt can also be run in a distributed fashion so that all heavy computation is performed on a remote machine where the data is stored and only the viewer and the GUI run on a local workstation.

4.1 Beam Selection

In order to identify the beam, i.e., find those particles that became accelerated, we first concentrate on the last timestep of the simulation (at $t = 37$). Using the parallel coordinates display, we select the particles of interest by applying a threshold of $px > 8.872 * 10^{10}$. As is visible in the parallel coordinates plot, those particles constitute two characteristic clusters in x direction (Figure 4 b). Using a pseudocolor plot of the data, we can then see the physical structure of the beam.

By tracing particles back in time, we see that the highest momentum and lowest momentum spread bunch observed in the simulation is formed at timestep $t = 27$ by the first bunch following the laser pulse (rightmost in these plots). In practice, the first beam following the laser pulse is typically the one of most interest to the accelerator scientists. The fact that the second beam shows higher or equal acceleration at the last time step of the simulation is due to the fact that the first beam will outrun the wave later in time and therefore switches into a phase of deceleration while the second beam is still in an acceleration phase. In practice, when researchers want to select only the first beam, they usually perform selection of the particles using thresholding in px at an

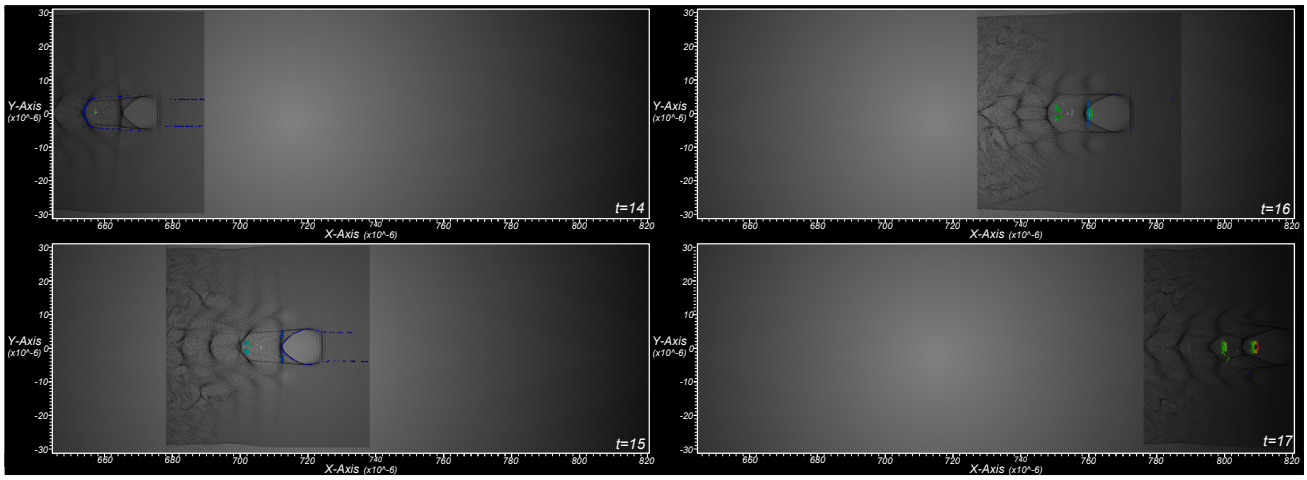

Fig. 5: Particles of the beam at timestep 14 (top left) to 17 (bottom right). The color of selected particles indicates x-momentum, px, while non-selected particles are shown in gray. We can readily identify two main sets of particles entering the simulation at timestep $t = 14$ and additional sets of particles entering at $t = 15$.

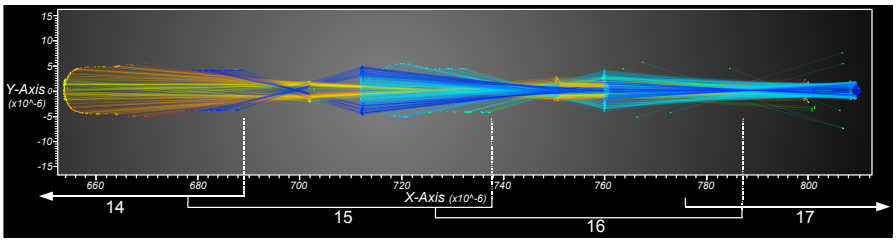

Fig. 6: The traces of the beam particles during timesteps $t = 14$ to $t = 17$ as shown in Figure 5. The positions of the selected particles at the different times are also indicated by annotations at the bottom of the figure. Here, we use color to indicate particle id. The begin and end of the timesteps in x direction are indicated below the bounding box.

earlier time, e.g. $t = 27$, rather than the last time step, here $t = 37$. By performing selection at an earlier time, one avoids selecting particles in the second beam while being sure to select all particles in the first beam. In this specific use case we are interested in analyzing and comparing the evolution of these two beams, which is why we performed selection at the last time step.

Figure 9(a and b) shows a similar example for the 3D particle dataset. At a much earlier time $t = 12$ ($x \approx 5.7 * 10^{-4}$ compared to $x \approx 1.3 * 10^{-4}$ in the 2D case) particles are trapped and accelerated, and the user selected particles in the first bunch via thresholding based on the momentum in x direction($px > 4.586 * 10^{10}$) and x position ($x > 5.649 * 10^{-4}$) to exclude particles in the secondary waves from the selection. In order to get an overview of the main relevant data, the user removed the background from the data first by applying a threshold of $px > 2.0 * 10^{9}$.

4.2 Beam Analysis

Having defined the beam in the 2D dataset, we can analyze formation of the beam by tracing the selected particles back to the time when particles entered the simulation and were then injected into the beam. In Figure 5, the individual particles of the beam are shown at timesteps $t = 14$ to $t = 17$. Here, color indicates the particles' momentum in the x direction (px). Figure 6 shows the

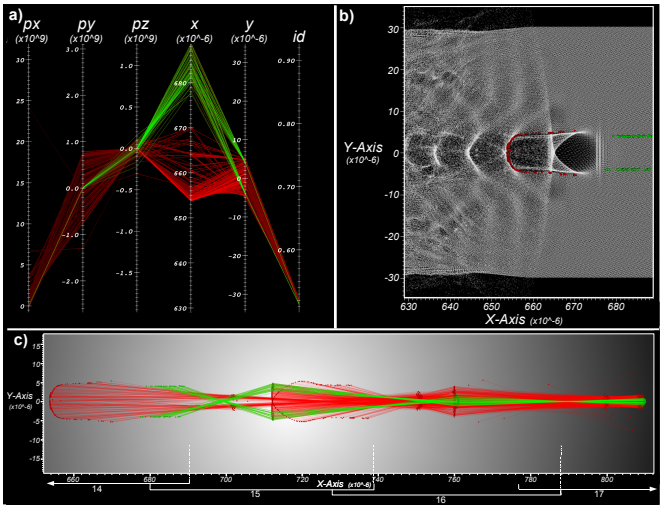

Fig. 7: a) By applying an additional threshold in x at timestep $t = 14$, we separate the two different set of particles entering the simulation. b) The refinement result, shown in physical space, includes all non-selected particles (gray) to provide context. c) Particle traces of the complete beam and the refined selection. In all plots we show the complete beam in red and the refined selection in green. After entering the simulation, the selected particles (green) define first the outer part of the first beam at timestep $t = 15$. Later on at timesteps $t = 16$ and $t = 17$, these particles become highly focused and define the center of the first beam.

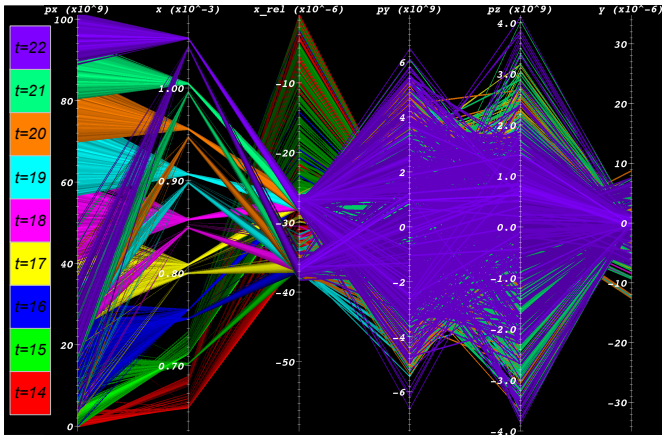

Fig. 8: The beam at timesteps $t = 14$ to $t = 22$ in a temporal parallel coordinates plots. Here, color indicates each of the discrete timesteps. We can readily identify the two different beams in x and x_{rel}. While the second beam shows equal to higher values in px during early time steps ($t = 14$ to $t = 17$), the first beam shows much higher acceleration at later times compared to the second beam.

particle traces over time colored according to particle ID's. Different sets of injection are readily visible, and two sets of particles appear at $t = 14$. The left bunch will be injected to form a beam in the second wake period, which is visible at $t > 14$. A second group of particles is just entering the right side of the box (recall that the simulation box is sweeping from left to right with the laser, so that plasma particles enter it from its right side). This bunch continues to enter at $t = 15$, and the particles stream into the first (rightmost) wake period. These particles are accelerated and appear as a bunch in the first bucket for $t > 15$. At the following timesteps $t = 16$ and $t = 17$, further acceleration of the two particle beams can be seen while only a few additional particles are injected into the beam, and these are less focused.

Based on the information at timestep $t = 14$, we then refine our initial selection of the beam. By applying an additional threshold in x, we can select those particles of the beam that are injected into the first wake period behind the laser pulse (see Figure 7a and 7b. By comparing the temporal traces of the selected particle subset (green) with the traces of the whole beam (red), we can readily identify important characteristics of the beam (see Figure 7c). After being injected, the selected particle subset (green) first defines the outer part of the first beam at timestep $t = 15$, while additional particles are injected into the center of the beam. Later on at timesteps $t = 16$ and $t = 17$, the selected particles become strongly focused and define the center of the first

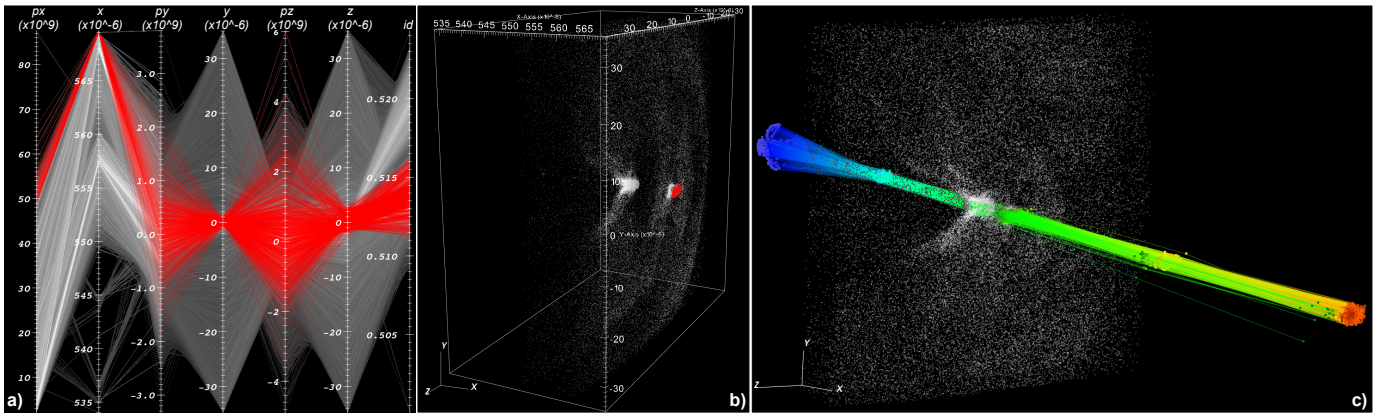

Fig. 9: a) Parallel coordinates of timestep $t = 12$ of the 3D dataset. Context view (gray) shows particles selected with $px > 2 * 10^9$. The focus view (red) consists of particles selected with $px > 4.856 * 10^{10}$ && $x > 5.649 * 10^{-4}$, which indicates particles forming a compact beam in the first wake period following the laser pulse. b) Pseudocolor plot of the context and focus particles. c) Traces of the beam. We selected particles at timestep $t = 12$, then traced the particles back in time to timestep $t = 9$ when most of the selected particles entered the simulation window. We also trace the particles forward in time to timestep $t = 14$. In this image, we here use color to indicate px. In addition to the traces and the position of the particles, we also show the context particles at timestep $t = 12$ in gray to illustrate where the original selection was performed. We can see that the selected particles are constantly accelerated over time (increase in px).

beam. By refining selections based on information at an earlier time, we are able to identify characteristic substructures of the beam.

Using temporal parallel coordinates, we can analyze the general evolution of the beam in multiple dimensions (see Figure 8). Along the x axis, two separate beams can be seen at all time steps ($t = 14$ to $t = 22$) with a quite stable relative position in x (x_{rel}). At early time steps, both beams show similar acceleration in px while later on, at timestep $t = 18$ to $t = 22$ (magenta to lilac), the particles of the first beam show significantly higher acceleration with a relatively low energy spread. Particles with a relatively high relative position in x direction are found mainly at timesteps $t = 14$ and $t = 15$ due to the fact that new particles enter the simulation window mainly during these times.

Figure 9c shows an example for the 3D dataset. Here, the traces of the particles selected earlier in Figure 9a are shown. We selected the particles at timestep $t = 12$ then traced them back to timestep $t = 9$ where most of the selected particles enter the simulation window and forward in time to timestep $t = 14$. As one can see in the plot, the selected particles are constantly accelerated over time.

5 Performance Evaluation

In this section, we present results of a study aimed at characterizing the performance of our implementation under varying conditions using standalone, benchmark applications. These unit tests reflect the different stages of processing we presented earlier in Section 4, the Use Case study. First, we examine the serial performance of histogram computation in Section 5.1: histograms serve as the basis for visually presenting data to a user via a parallel coordinates plot. Second, in Section 5.2, we

examine the serial performance of particle selection across all time steps of simulation data. Finally in Section 5.3, we examine the parallel scalability characteristics of our histogram computation and particle tracking implementations on a Cray XT4 system.

For the serial performance tests in Sections 5.1 and 5.2, we use the 3D dataset described earlier in Section 4: 30 timesteps worth of accelerator simulation data, each timestep has about 90 million particles and is about 5*GB* in size. The aggregate dataset size is about 210*GB*, including the index data. We store and retrieve simulation and index data using HDF5 and a veneer library called *HDF5-FastQuery* [9]. HDF5-FastQuery presents an implementation-neutral API for performing queries and obtaining histograms of data. We conduct the serial performance tests on a workstation equipped with a 2.2Ghz AMD Opteron CPU, 4GB of RAM and running the SuSE Linux distribution.

The serial performance tests in Sections 5.1 and 5.2 measure the execution time of two different implementations that are standalone applications we created for the purpose of this performance experiment. One application, labeled "FastBit" in the charts, uses FastBit for index/query and histogram computation. The other, labeled "Custom" in the charts, does not use any indexing structure, and therefore performs a sequential scan of the dataset when computing histograms and particle selections.

5.1 Computing 2D Histograms

We run a pair of test batteries aimed at differentiating performance characteristics of computing both unconditional and conditional histograms. The unconditional histogram is simply a histogram of an entire dataset using a set of application-defined bin boundaries. A conditional histogram is one computed from a subset of data records that match an external condition.

5.1.1 Unconditional Histograms

In our use model, the computation of the unconditional histogram is a "one-time" operation. It provides the initial context view of a dataset to the user. Since this computation requires examining all data records, we expect little, if any, performance variation of the serial implementation as we vary the number and resolution of histogram bins.

For this test, we vary the number of bins in a 2D histogram over the following bin resolutions: 32x32, 64x64, 128x128, 256x256, 512x512 and 1024x1024 bins. All histograms span the same range of data values: increasing the bin count results in bins of finer resolution. The results of this test are shown in Figure 10. Since both the FastBit and Custom applications need to look at all the data records to compute an unconditional histogram, we do not expect much performance variation as we change the number of bins. FastBit is generally faster than the custom code throughout primarily because of the difference in organization of the histogram bin counts array. FastBit uses a single array, which results in a more favorable memory access pattern.

5.1.2 Conditional Histograms

In contrast to the unconditional histogram, which is a one-time computation, the process of visual data exploration and mining will rely on repeated computations of conditional histograms. Therefore, we are particularly interested in achieving good performance of this operation to support interactive visual data analysis. This set of tests focuses on a use model a change in the set of conditions will result in examining a greater or lesser number of data records to compute a conditional histogram. This set of conditions reflects the repeated refinement associated with interactive, visual data analysis.

We parameterize our test based on the number of hits resulting from a range of different queries. We use thresholds of the form

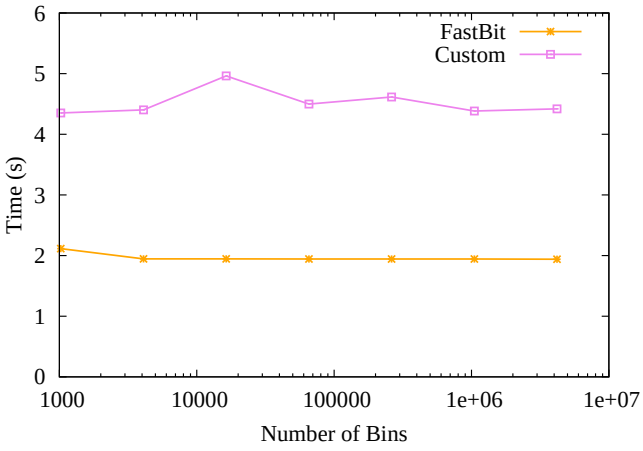

Fig. 10: Timings for serial computation of unconditional histograms.

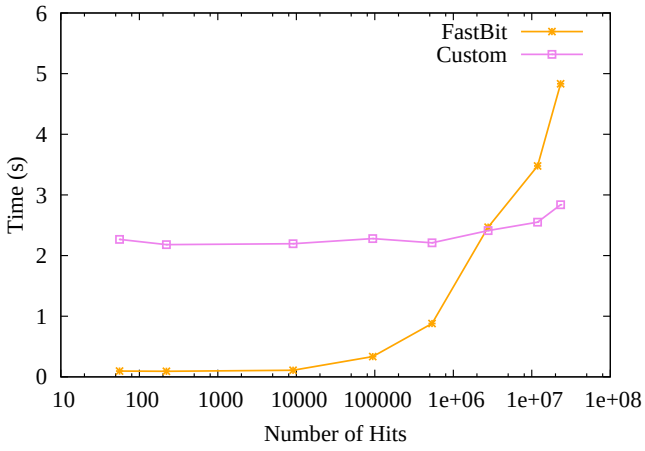

Fig. 11: Timings for serial computation of conditional histograms.

$px > ...$ as the histogram conditions. As we increase the threshold of particle momentum in the x direction with this condition, fewer data records (particles) match that condition and contribute to the resulting histogram. In these tests, we hold the number of bins constant at 1024x1024.

Figure 11 presents results for computing conditional histograms using FastBit and the Custom application. We observe that for small number of hits, the FastBit execution times are dramatically faster than the Custom code, which is forced to examine all data records. It is important to note that typical visual analysis queries typically isolate a small number of particles – from tens to thousands – and FastBit provides outstanding performance in this regime. As the number of hits increase, approaching 1M and 10M+ particles – which is a significant fraction of the 90M total particles – a sequential scan through all the data records produces better results. This is due to the fact that FastBit copies the hits to an intermediate array and computes histograms from that array. When the intermediate array size exceeds the size of the cache, accessing the array becomes expensive. This procedure could be further optimized by removing the intermediate array.

5.2 Particle Selections

Following our use model, once a user has determined a set of interesting data conditions, like particles having a momentum exceeding a given threshold, the next activity is to extract those particles from the large dataset for subsequent analysis. This set of tests aims to show performance of the particle subsetting part of the processing pipeline.

The execution time for this task is clearly proportional to the size of the selection – the time required to find a set of particles in a large, time-varying dataset varies as a function of the size of the particle search set as well as the size of the simulation data itself. We parameterize the search set size for this set of performance tests, varying the number of particles to search for over values ranging from 10, 100, 1000, ... , up to 20M particles.

Our custom code uses a sequential scan of the entire dataset to search for particles in the search set. For each data record, it compares the particle ID of the record to the search set using an efficient algorithm: if the size of the search set is S, then the search time is $log(S)$. If there are N data records in the entire dataset, the computational complexity of the entire algorithm is

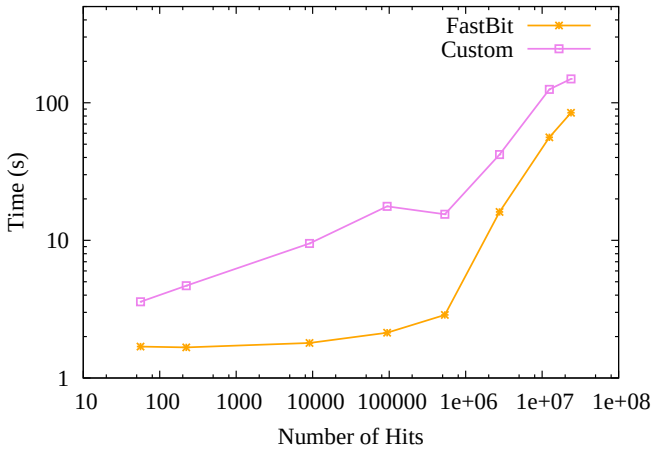

Fig. 12: Timings for serial processing of identifier queries.

$Nlog(S)$.

Figure 12 presents results for running ID queries using FastBit and the Custom code for one timestep. We observe that FastBit demonstrates very good performance for most queries of interest where we are isolating particles numbering from tens to thousands. With an increasing number of particles, the performance of both FastBit and the custom code degrades.

5.3 Scalability Tests

Whereas the previous sections focus on the serial performance of computing histograms and performing particle subset selections, this section focuses on the scalability characteristics of both algorithms.

The platform for these tests is `franklin.nersc.gov`, a 9,660 node, 19K core Cray XT4 system. Each of the nodes consists of a 2.6Ghz, dual-core AMD Opteron processor and has 4GB of memory and runs the Compute Node Linux distribution. One optimization we use on this machine at all levels of parallelism is to restrict operations to a single core of each node. This optimization will maximize the amount of memory and I/O bandwidth available to each process in our parallel performance tests. On this platform, the Lustre Parallel Filesystem serves data to each of the nodes. The nodes used in the study are a small fraction of a larger shared facility with a dynamic workload. Our scalability tests cover parallelism levels over the following range: 1, 5, 10, 30, 50, 75 and 150 nodes of the Cray XT4.

As in the previous section, we report wall-clock times which encapsulate CPU processing and I/O. We report speedup factor as the ratio of time taken by a single node to the times taken by the node subset to complete a task. We do a strong scaling test by keeping the problem size fixed at 150 timesteps for all cases.

The dataset used in this study totals about $1TB$ in size. This dataset is the same 3D dataset described earlier in Section 4, though we replicated each individual time steps five times to achieve a total dataset size about five times larger than that reported in Section 4[3]. We employ a fairly simple form of domain decomposition for these scalability tests: namely we assign subsets

[3]Note to reviewers: our accelerator science colleagues on this paper presently have jobs in Franklin's queue that will result in a dataset that exceeds the 1TB threshold. We were not able to complete these runs prior to the SC08 paper deadline. Our plan is to rerun the scalability study with the new dataset once it has been generated and use those charts in the final version of this paper should it be accepted for the SC08 technical program. We do not expect any difference in results of the scalability study since the distribution of particle values (momentum, position, etc.) will not be substantially different than that of the dataset with

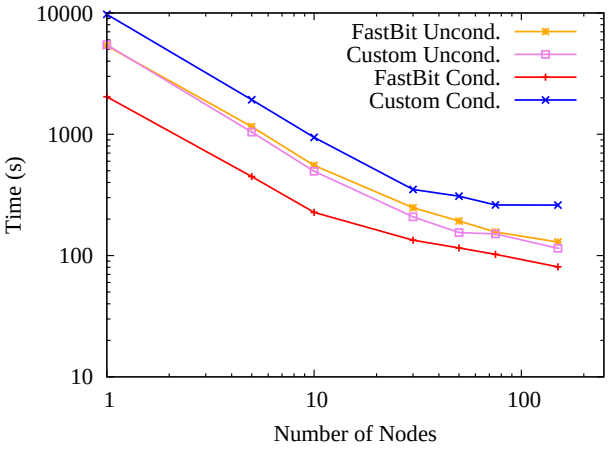

Fig. 13: Timings for parallel computation of histograms.

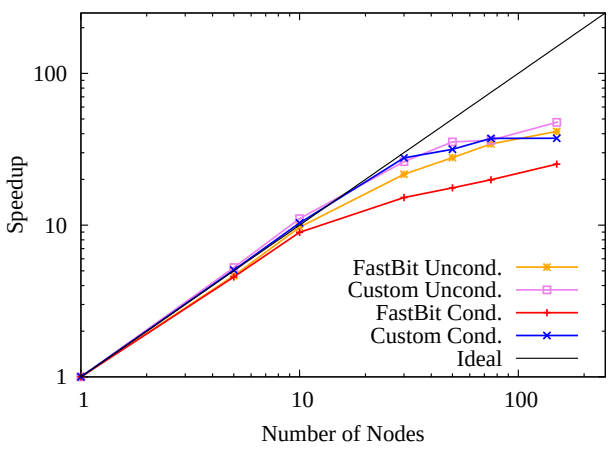

Fig. 14: Scalability of parallel histogram computation.

of timesteps (corresponding to individual HDF5 files) to individual nodes for processing. The subsets are statically assigned to nodes in a strided fashion.

Following the the theme of use cases reported in previous sections, we report times for realistic science usage scenarios. For histogram computation, we generate five parallel histogram pairs for the position and momentum fields. We use 1024x1024 bins, which is a reasonable upper limit given typical screen resolutions. For conditional histograms, we use $px > 10^{10}$ query; for particle tracking, we report numbers for $px > 9 * 10^{10}$ query, which results in 10K hits. All of these choices are grounded in discussions with our science collaborators and reflect reasonable ranges and thresholds.

Figure 13 presents results from parallelizing the computation of both conditional and unconditional histograms over multiple nodes. As expected, we observe that the computation time decreases as we add more nodes. Similar to the serial case, we do not observe much of a difference between FastBit and the Custom application for computing the unconditional histogram since both implementations will examine all data records. For computing the conditional histogram, FastBit maintains its advantage over the Custom application. Figure 14 presents the speedup factors corresponding to this computation; we observe a very favorable speedup. This is to be expected since the nodes can perform their computations independent of others.

Figure 15 presents results from particle tracking over 150 timesteps. Similar to the serial case, FastBit is faster than the Custom application, and maintains its advantage when run in parallel. Figure 16 demonstrates that we achieve excellent scalability in parallelizing this computation. We observe super-linear speedup when using 75 and 150 nodes. This result is due to the fact that nodes are able to cache the variables of interest in local memory and do not have to go to the Lustre filesystem to access data. We observe that when using 150 nodes, FastBit is able to track 10K particles over 1TB of data in as little as 2 seconds. This marks a dramatic improvement over the current scripts used by our science collaborators: what formerly required over 2 hours to track 300 particles in the 210GB dataset on a SMP machine now takes seconds.

timestep replication

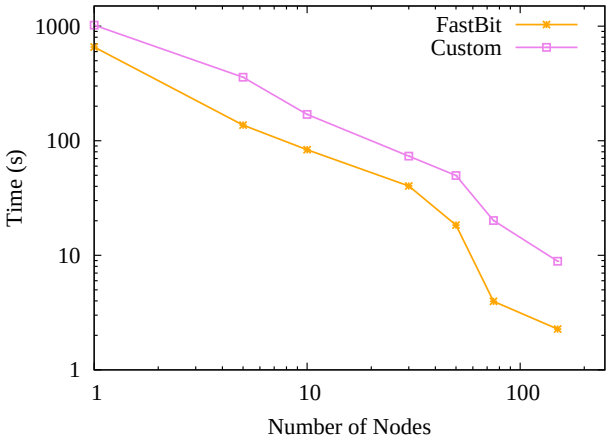

Fig. 15: Timings for parallel particle tracking.

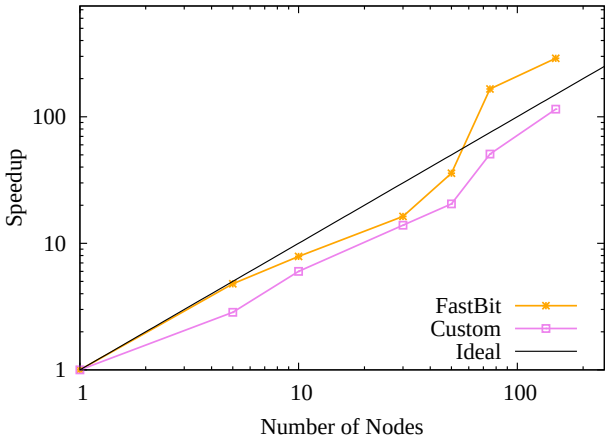

Fig. 16: Scalability of parallel particle tracking.

6 Conclusions and Future Work

Managing and gaining insight from vast amounts of data is widely accepted as one of the primary bottlenecks in modern science. The work we present takes aim at enabling rapid knowledge discovery from large, complex, multivariate and time-varying scientific datasets and using modern HPC platforms. To achieve this objective, we have presented a novel approach for quickly creating histogram-based parallel coordinates displays. Further, we leverage this form of visual information display as the basis for forming complex multivariate range queries. We combined this form of visual information display with state-of-the-art index/query technology to quickly compute conditional histograms as well as to quickly and efficiently extract subsets of data that meet a set of conditions. We presented a case study showing how these technologies are used in concert to explore a large, time-varying dataset produced by a laser wakefield accelerator simulation. That exploration reveal interesting beam characteristics, such as how particles are first accelerated by the wakefield, then later they "outrun" the wave and undergo a period of deceleration.

We also conducted a performance study of each of the fundamental algorithms that form a complete implementation in a production-quality, parallel capable visual data analysis application. That study shows the efficacy of our algorithms for computing histograms and for performing particle subset selection. These algorithms were shown to have favorable scalability characteristics over a set of processor pool sizes ranging from 1 up to 150 on a modern HPC platform, a Cray XT4. From the scientists' point of view, they are now able to perform a type of visual data analysis in a few seconds with this new work as compared to the several hours required using legacy tools. These results are significant because they show the successful synergy that can result when combining visual data analysis with scientific data management technologies to solve challenging problems in scientific knowledge discovery.

As we move forward with this work, we are particularly interested in exploring different avenues for parallelizing the most expensive parts of the visual data analysis work. As visual data analysis is typically an interactive process, minimizing the response time is crucial to maximize its effectiveness. We also plan to apply this system to larger datasets from this and other scientific disciplines. As our implementation already supports a mode of operation whereby a parallel back end runs on the HPC platform to perform I/O, visualization and analysis processing with results being transmitted over the network to a client running

on a workstation, we are in a good position to tackle such future challenges. Finally, the work we show here is primarily visual data analysis; we intend to extend this work to couple it with more traditional data analysis techniques.

ACKNOWLEDGEMENTS

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET). This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

[1] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[2] E. W. Bethel, S. Campbell, E. Dart, K. Stockinger, and K. Wu. Accelerating Network Traffic Analysis Using Query-Driven Visualization. In *Proceedings of 2006 IEEE Symposium on Visual Analytics Science and Technology*, pages 115–122. IEEE Computer Society Press, October 2006. LBNL-59891.

[3] C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. In *SIGMOD*, pages 355–366, 1998.

[4] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, Mar. 1997.

[5] H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max. A contract-based system for large data visualization. In *Proceedings of IEEE Visualization 2005*, pages 190–198, October 2005.

[6] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *IEEE Visualization 1999*, pages 43–50, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[7] C. Geddes, C. Toth, J. van Tilborg, E. Esarey, C. Schroeder, D. Bruhwiler, C. Nieter, J. Cary, and W. Leemans. High-Quality Electron Beams from a Laser Wakefield Accelerator Using Plasma-Channel Guiding. *Nature*, 438:538–541, 2004. LBNL-55732.

[8] C. G. R. Geddes. *Plasma Channel Guided Laser Wakefield Accelerator*. PhD thesis, University of California, Berkeley, 2005.

[9] L. Gosink, J. Shalf, K. Stockinger, K. Wu, and E. W. Bethel. HDF5-FastQuery: Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*. IEEE Computer Society Press, July 2006. LBNL-59602.

[10] S. Guha, K. Shim, and J. Woo. Rehist: relative error histogram construction algorithms. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large Data Bases*, pages 300–311. VLDB Endowment, 2004.

[11] A. Inselberg. Parallel coordinates for multidimensional displays. In *Spatial Information Technologies for Remote Sensing Today and Tomorrow, The Ninth William T. Pecora Memorial Remote Sensing Symposium*, pages 312–324. IEEE Computer Society Press, 1984.

[12] A. Inselberg. *Parallel Coordinates Visual Multidimensional Geometry and Its Applications*. Springer-Verlag, 2008.

[13] A. Inselberg, H. Hauser, M. Ward, and L. Yang. Modern parallel coordinates: from relational information to clear patterns, tutorial. In *IEEE Visualization*, October 2006.

[14] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing structure within clustered parallel coordinates displays. In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 17, Washington, DC, USA, 2005. IEEE Computer Society.

[15] C. Nieter and J. R. Cary. VORPAL: A Versatile Plasma Simulation Code. *J. Comput. Phys.*, 196(2):448–473, 2004.

[16] M. Novotný. Visually effective information visualization of large data. In *Proceedings of Central European Seminar on Computer Graphics (CESCG)*, 2004.

[17] M. Novotný and H. Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006.

[18] P. O'Neil. Model 204 architecture and performance. In *2nd International Workshop in High Performance Transaction Systems, Asilomar, CA*, volume 359 of *Lecture Notes in Computer Science*, pages 40–59. Springer-Verlag, Sept. 1987.

[19] B. W. Silverman. *Density Estimation for statistics and data analysis*. Monographs on Statistics and Applied Probability 26. Chapman and Hall, London, 1986.

[20] K. Stockinger, E. W. Bethel, S. Campbell, E. Dart, , and K. Wu. Detecting Distributed Scans Using High-Performance Query-Driven Visualization. In *SC '06: Proceedings of the 2006 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, October 2006.

[21] K. Stockinger, E. W. Bethel, S. Campbell, E. Dart, and K. Wu. Detecting Distributed Scans Using High-Performance Query-Driven Visualization. In *SC '06: Proceedings of the 2006 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA, October 2006. IEEE Computer Society Press. LBNL-60053.

[22] K. Stockinger, J. Shalf, K. Wu, and E. W. Bethel. Query-Driven Visualization of Large Data Sets. In *Proceedings of IEEE Visualization 2005*, pages 167–174. IEEE Computer Society Press, October 2005. LBNL-57511.

[23] E. J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, Sept. 1990.

[24] K. Wu, W. Koegler, J. Chen, and A. Shoshani. Using bitmap index for interactive exploration of large datasets. In *SSDBM'2003*, pages 65–74, Washington, DC, USA, 2003. IEEE Computer Society.

[25] K. Wu, E. Otoo, and A. Shoshani. Compressing bitmap indexes for faster search operations. In *SSDBM'02*, pages 99–108, Edinburgh, Scotland, 2002.

[26] K. Wu, E. Otoo, and A. Shoshani. On the performance of bitmap indices for high cardinality attributes. In *VLDB*, pages 24–35, 2004.

[27] K. Wu, E. Otoo, and A. Shoshani. Optimizing bitmap indices with efficient compression. *ACM Transactions on Database Systems*, 31:1–38, 2006.